Embedded analytics delivers system-wide visibility for debug, safety, security and more...

Design and Reuse IP-SoC Days – Shanghai 2017

Agenda

- Some obvious statements
- Some problems with existing approaches
- Key requirements
- The UltraSoC approach
- Some examples of performance analysis and debug
- Use cases
- Summary

Some obvious statements

- SoCs have become increasingly complicated & are not going to get simpler
  - Contain several processors, from different vendors
    - Verified in isolation and come with test suite
  - Contain 100s of IP blocks
    - Each verified in isolation
  - Contain complex interconnects
    - Verified for certain, identified conditions
  - Software created by large disparate teams
    - If lucky, modules and subsystem verified for certain, identified conditions.
  - All this has to successfully work together
- Understanding real world system behaviour is HARD!

Some problems with existing approaches

- Processor-centric, not system-centric
  - Processors are a very small part of the overall system
- It's very difficult to monitor:
  - Bus behaviour, memory controllers, interactions between blocks
- There is very little analytics
  - Just extracting raw data
- Intrusive
- Ad hoc
- Developing, but still essentially signal-based
  - Hard to close timing
- In-field monitoring is not easy

Key requirements

- A system-centric vendor-neutral debug and monitoring infrastructure
  - One that enables access to different proprietary debug schemes
  - Enables monitoring of interconnect, interfaces and custom logic
  - Run-time configurable
    - Re-use the hardware to provide visibility for different scenarios
    - Run-time configuration of cross-triggering
    - Support 10s if not 100s of cross-triggering events
  - These can be interrogated after a problem to determine actual status
  - Need to be power aware
  - Built-in security
  - Can be used during the whole development flow and in the field

UltraSoC embedded analytics architecture

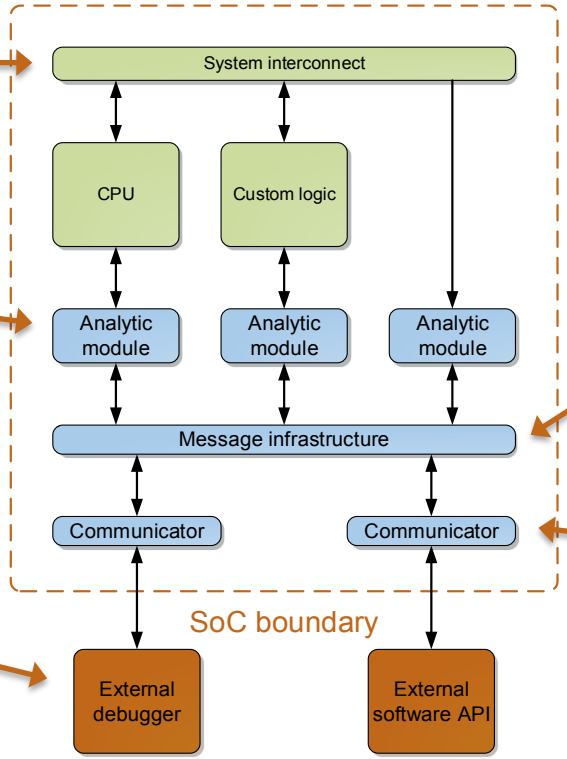
System modules and interconnect

1. Protocol-aware analysis modules with “smart” filters and trace

2. Optimized message-passing infrastructure

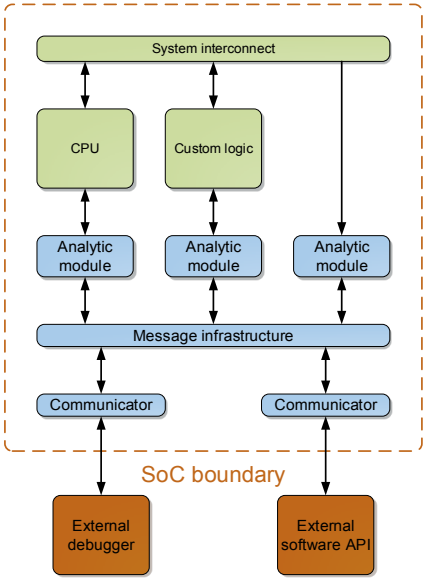
3. Communicators. Eg: USB, JTAG, streaming, on-chip

4. Visualization software

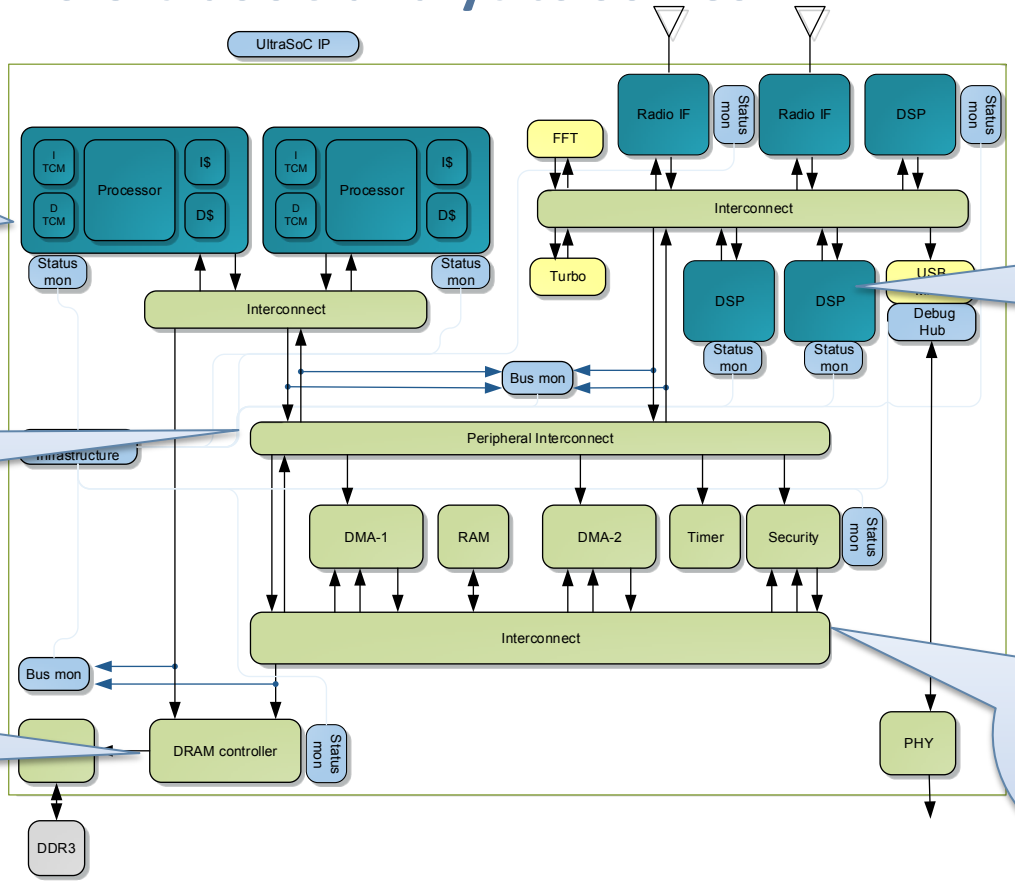


How does it work?

- Protocol-aware analysis modules
  - Processors: ARM, MIPS, Ceva, RISC-V, + more
  - Buses: AXi, CHI, Netspeed, + more
- Filter, match, trigger, store, output
  - Analysis done in hardware, on-chip
  - Reduces need for high-speed off-chip transport
  - Can be used in-system and in-field
- A choice of communicators
  - To suite system requirements



Example problems UltraSoC analytics solves



Why is the CPU not performing as fast as expected?

What is the mismatch between the host & the DSP?

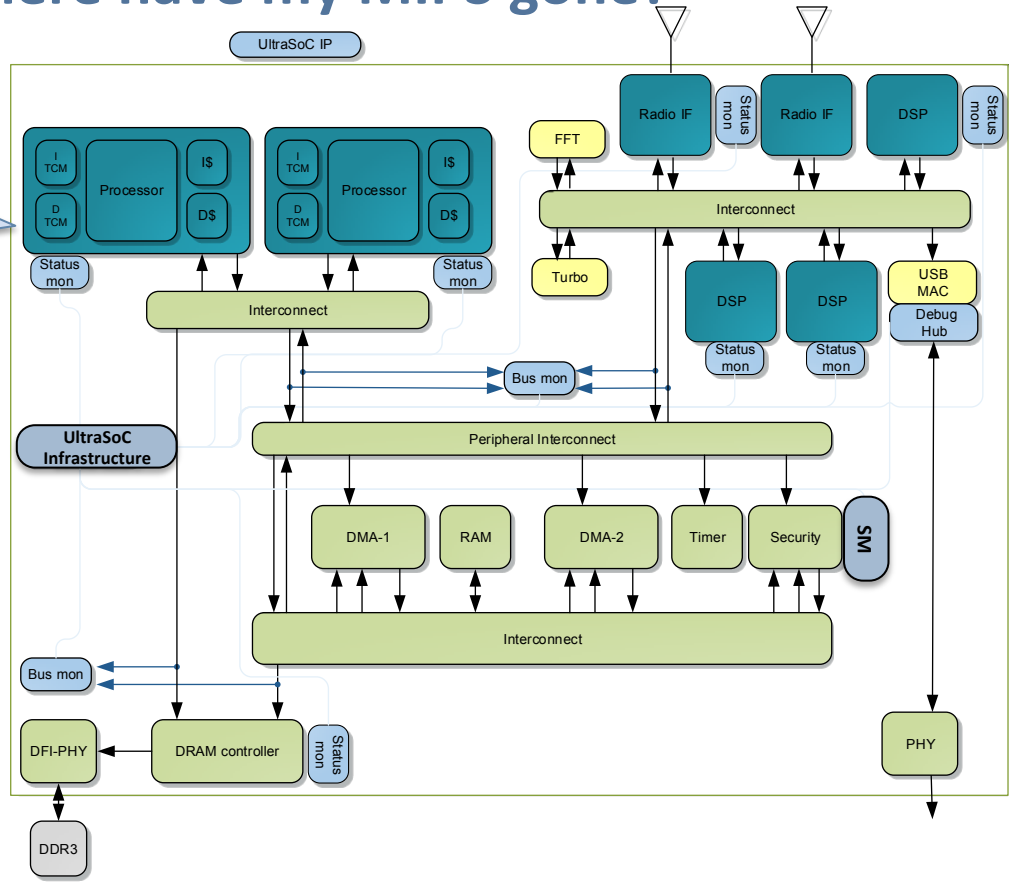
Why do some DMA transfers take too long?

What is going on with my memory controller?

Why does the system hang or deadlock on rare occasions?

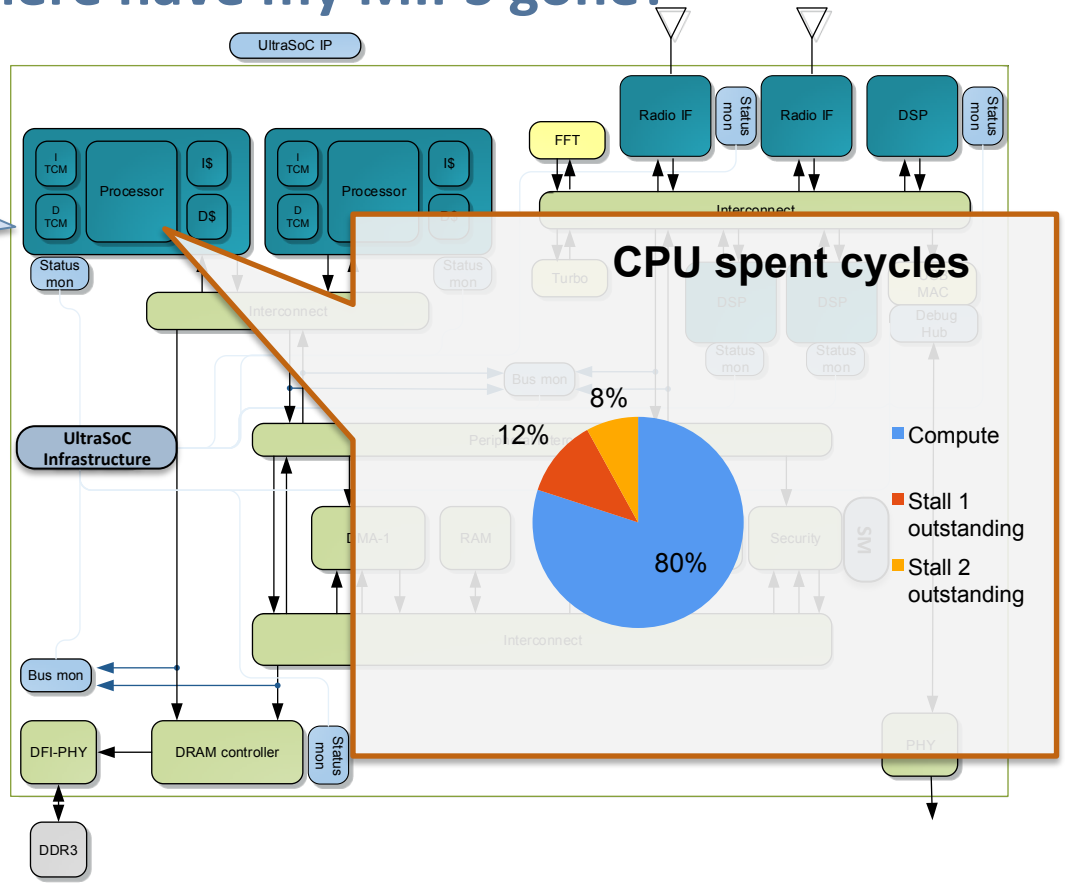
Example 1: “Where have my MIPS gone?”

Why is the CPU not performing as fast as expected?

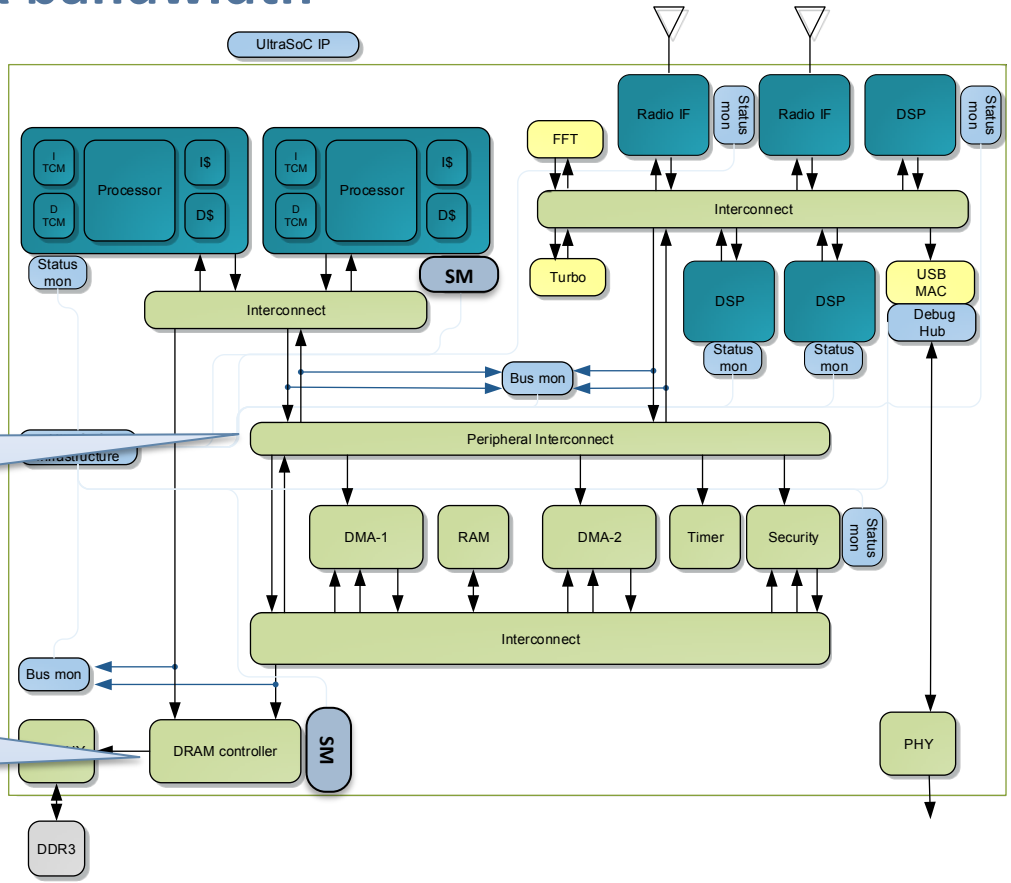


Example 1: “Where have my MIPS gone?”

Why is the CPU not performing as fast as expected?



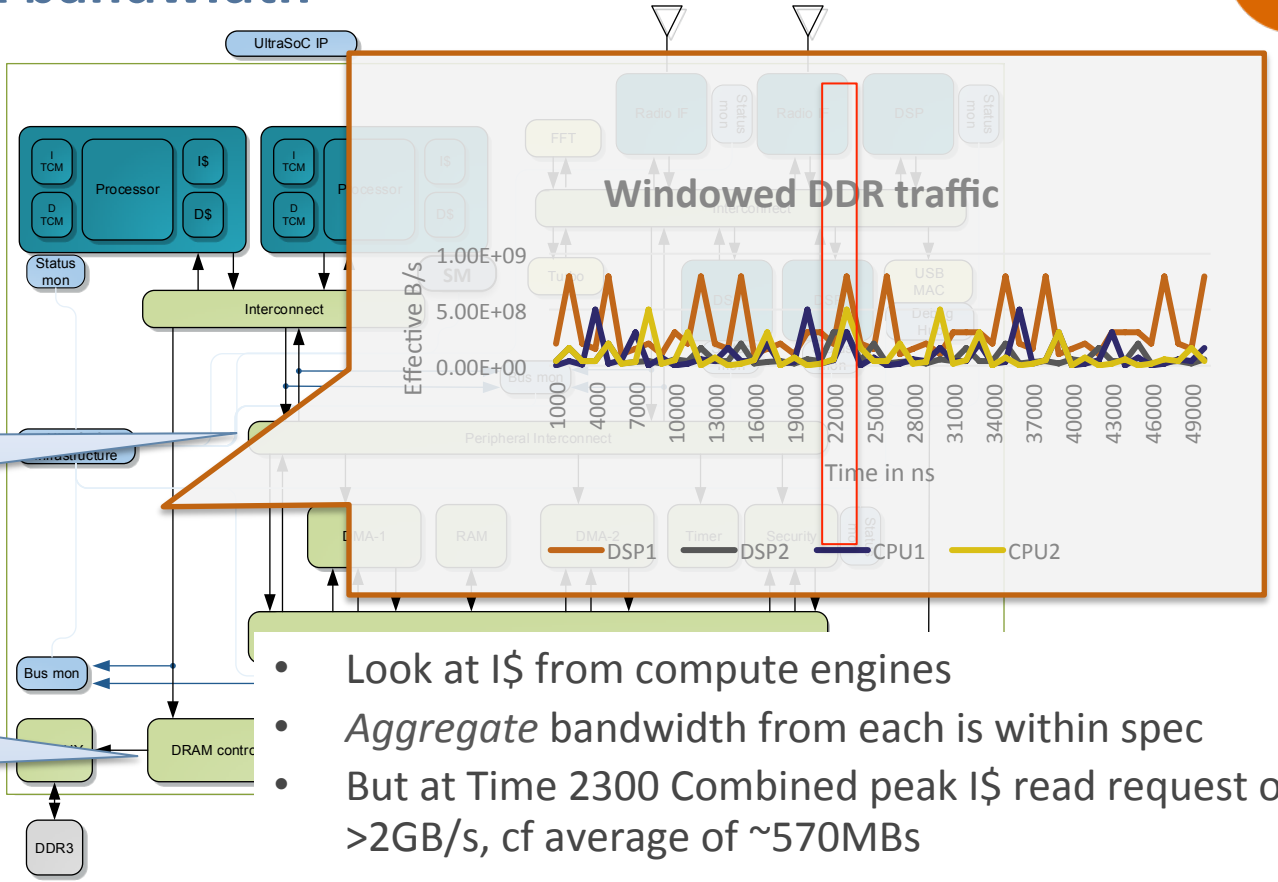
Example 2: DDR bandwidth



Why do some DMA transfers take too long?

What is going on with my memory controller?

Example 2: DDR bandwidth

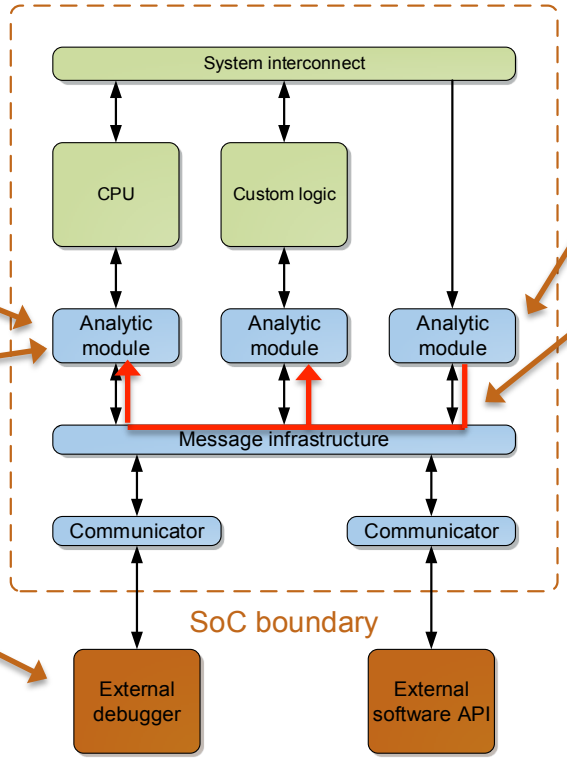


Why do some DMA transfers take too long?

What is going on with my memory controller?

- Look at I$ from compute engines
- *Aggregate* bandwidth from each is within spec
- But at Time 2300 Combined peak I$ read request of >2GB/s, cf average of ~570MBs

Cross-triggering



1. If write to an address range occurs, send event A

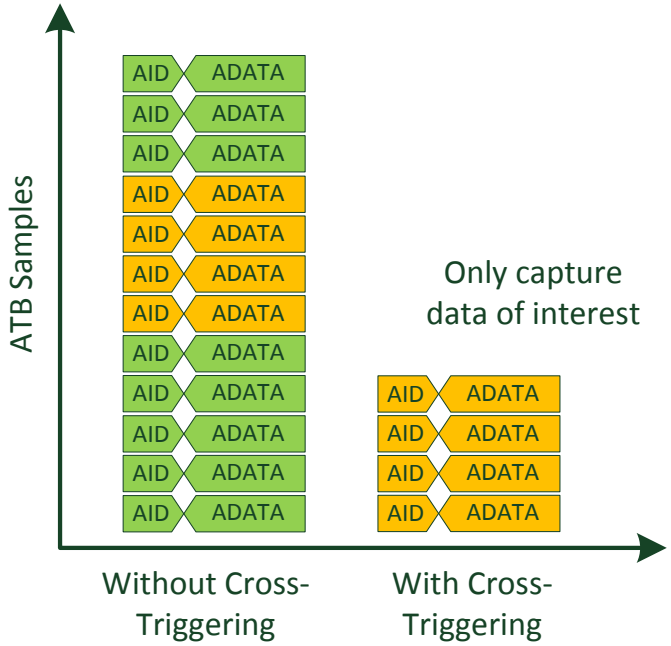
2. Configure to store trace in ring buffer. Stop & output on event A

3. Write detected. Event A broadcast to all modules in chip

4. Event A received. Output trace

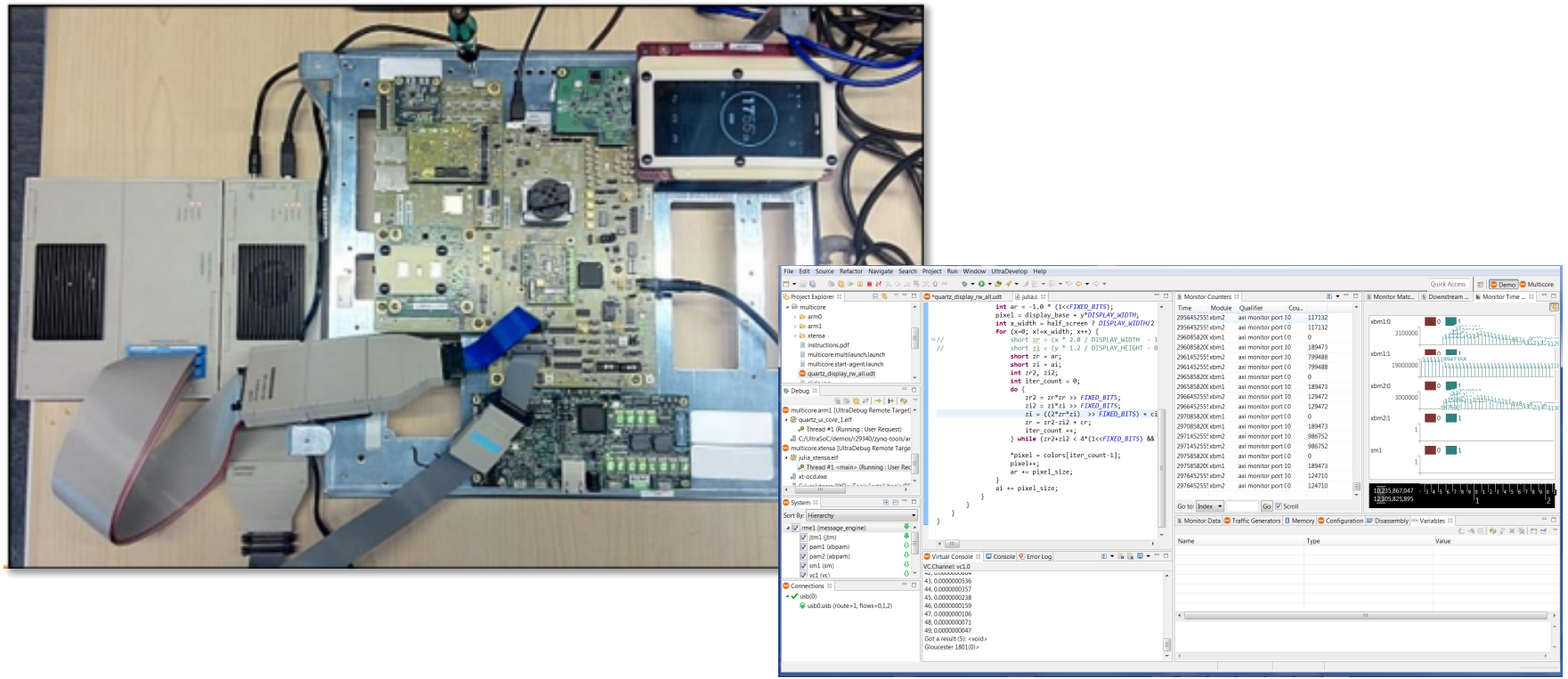
5. Trace data displayed

The importance of cross-triggering

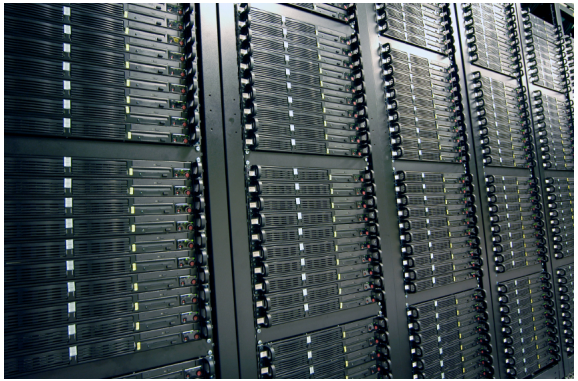


- Gigabytes of trace data can be reduced to kilobytes
- In-field, events that only occur once a week can be captured and uploaded
- Cross-trigger events can be sourced from anywhere, even hardware signals
- Run-time selection is essential

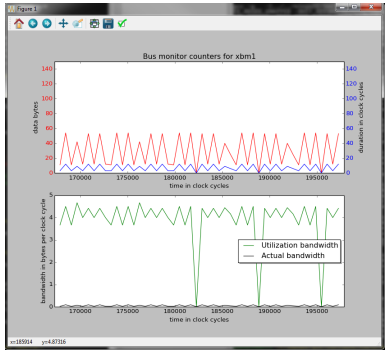
Use case 1: classic debug



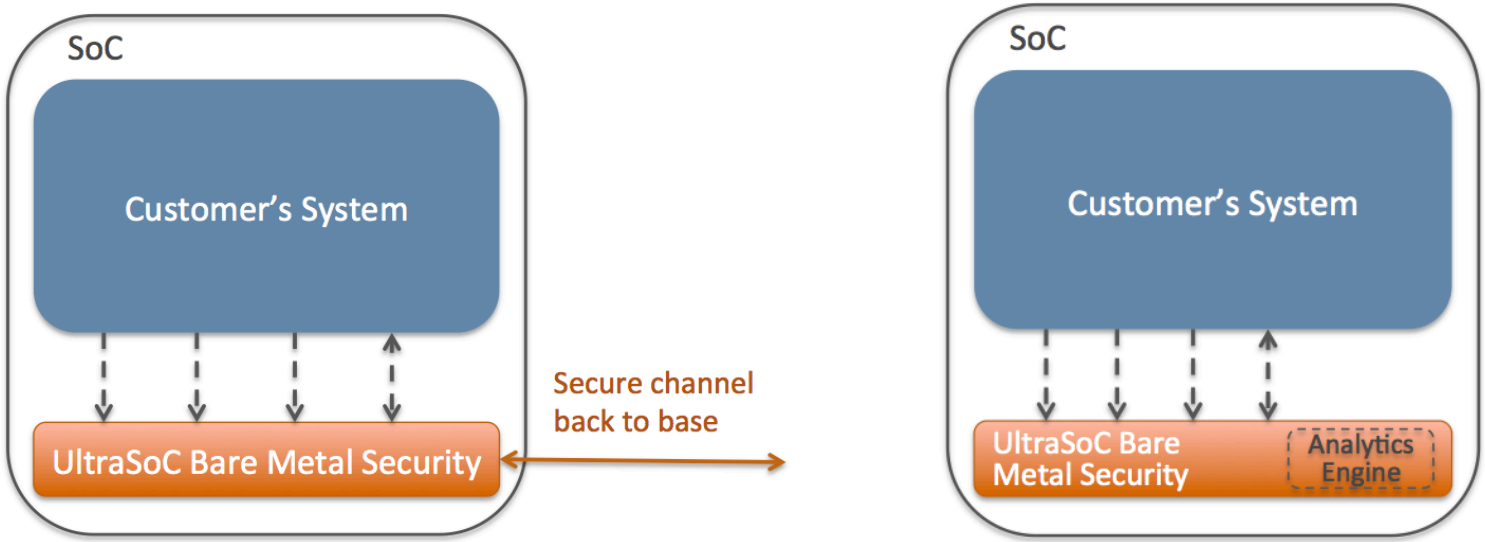
Use case 2: in-field debugging and analysis



- Find the cause of rate problems
- Monitor ongoing performance
- Fix problems through upgrades
- Input to next-generation SoC



Use-case 3: bare metal security and safety



Summary

- In complex SoCs
  - Embedded analytics is essential
  - A unified approach can save months of effort and a lot of money
- Embedded analytics hardware can be used for
  - Classic lab debug
  - In field problem solving
  - Lifetime analysis
  - A separate domain to enhance security and safety